Electronic Books: Definition, Genres, Interaction Design Patterns

Jan O. Borchers
Linz University, Austria
+49 731 502 4192
jan@tk.uni-linz.ac.at

April 2, 1999

Abstract

This paper summarizes the author's positions on a number of user interface design issues for electronic books (e-books). It proposes a definition and genre classification for e-books, and suggests feature priorities for each genre. It also describes a pattern language approach adopted from architecture to express user interface design guidelines for e-books.

1 Introduction

The goal of this position paper is to summarize three kinds of information: significant past research results by the author and others, with appropriate citations given, in order to set the field, and preliminary results that need further verification, but that may spark new thoughts, discussions, and research initiatives through the workshop.

This text refers to electronic books in the sense of the following definition:

> *Definition: Electronic Book.* An electronic book, or e-book, is a portable hardware and software system that can display large quantities of readable textual information to the user, and that lets the user navigate through this information.

The terms used in this definition have the following rationale:

e-book: This introduces an abbreviation for *electronic book*.

portable: This excludes, e.g., a web browser on a desktop PC.

hardware and software: This excludes reader software and specific eBook "titles". On the other hand, PDAs etc. can also be e-books.

display: This focuses the definition on devices for visual presentation (although audio is a useful add-on).

large quantities: This requires that more than one or a few lines of text can be displayed, and excludes, e.g., digital watches or pagers.

readable textual information: This emphasises text as primary contents, and excludes devices such as special-purpose numeric data displays.

navigate: Just displaying a single, static page is not enough either.

2 A genre classification for e-books

Traditional books can be classified into genres according to various criteria. The most useful classification, the author believes, follows the idea of user-centered design and is according to the primary *task* that the user of a document is aiming to perform.

We have proposed a similar classification for electronic kiosk systems several years ago [8]. It has subsequently proven very useful on a number of occasions where design goals needed to be discussed. There, we classified such systems into *information*, *service*, *advertisement* and *entertainment* kiosks. Few systems only have one of the above tasks as design goal, but usually one *primary* goal can be identified in this way.

Adler et al. [1] give a classification of work-related reading activities. They distinguish between reading for document identification, document skimming, to remind oneself, for reference (to answer questions), to inform oneself (newspaper), to learn, for cross-referencing (to integrate multiple sources), to edit or review, to support listening (to a talk, lecture or presentation), and to support discussion (handout in a group). Across a wide variety of professions, they found reading

for reference, cross-reference, and discussion support to be the most frequent tasks (over 70% of total reading & writing time), and they conclude that features of digital reading devices need to differ substantially for these different tasks.

However, their survey explicitly excludes reading outside the working environment. This results in linear and continuous, book-like reading tasks to be underrepresented, and in certain goals, like reading for entertainment, not to be taken into account at all. Some of the other styles, like reading to remind oneself of something (e.g., a to-do list), do not qualify as e-book genres.

For e-books, we therefore propose the following *genre classification:*

reference and documentation: This material is read to answer a concrete question. This genre includes dictionaries, phone books, or reference manuals.

learning: The goal is to take in information in a structure form for later application. Examples are tutorials, school books, and nonfiction in general.

browsing: Skimming a document to get an idea of its overall contents, and decide whether to read it in detail. Examples are newspapers, journals, and magazines. When an article seems interesting, this can turn into reading for learning.

entertainment: These books are read to relax and enjoy oneself. Examples are novels, or comics.

3 Advantage parameters

The document genre clearly influences the "features" of traditional books, such as physical size, binding style, paper type and quality, layout, etc. We state that this is also true for e-books.

In each genre, we identify a key *advantage parameter* which can vary between different document types within this genre. The parameter therefore further orders that genre, and it influences the degree to which a document can make good use of the features e-books can offer. This addresses the question "what activities benefit from such devices, and which are better left on paper."

reference and documentation: The key parameter is *linear reading probability*. This is lowest with books such as a dictionary, and highest with well-structured, readable, almost tutorial-like reference documentation. The more likely linear, sustained reading is, the more important display and handling quality become. An example: People readily use, and may even prefer, an electronic telephone dictionary over its paper equivalent, but they print out reference manuals as soon as they think they will read more than just some snippets of them.

learning: The key parameter is *self-containedness*. The more external references a text has, the more it benefits from electronic links to those sources. (At the same time, though, reading simultaneously in several sources spread out on the table is something that traditional paper supports much better than an e-book that only contains a singular display for many pages and book titles.)

browsing: The longer the *lifespan* of the document, the more advantages a paper version has. Example: The daily newspaper is discarded after one day, so its value as a physical document is regarded as quite low. A monthly journal, on the other hand, is usually archived by the user for a longer period, and is accordingly produced to be more durable and expensive.

entertainment: The higher the *complexity* of the contents, the more useful typical reference and annotation features such as cross-linking, full-text search, marking, etc. become: Few people will need to go back to re-read a certain passage for better understanding in a comic book, but this could become quite useful in Thomas Mann's *Zauberberg*.

In general, critique against e-books is directed mainly at those titles that represent "cultural value", i.e., prose, lyrics, or non-fiction. The e-book form is already widely accepted for non-cultural titles (e.g., phone books). Also, cultural value is usually correlated with parameters such as sustained reading time.

From a user-oriented point of view, it is also important to take into account that many people like to *display* the book titles they own and have (or will) read.

4 Advantages of e-books: Attempting a summary

In terms of usability, the following points are often given as advantages for electronic books (for example, [15]). This list may serve as a starting point of a comprehensive comparison of paper and electronic media. It is useful, however, to consider not only the end user, but different target groups and examine their specific

a critical value around 150dpi that needs to be exceeded to make e-books usable. Even higher resolution is necessary for simultaneous overview and detail.

contrast and brightness: These are still far better on printed media. However, digital ink as being developed in the Gyratron and E-Ink projects, may be able to improve these factors for electronic devices.

colour: At least for portable devices such as e-books, colour intensity and colour ranges are not of the quality of printed documents. However, since a large part of books appears in black-and-white print even today, this factor is not as important as other display attributes.

The form factor

weight: Even though an e-book is easier to carry than a dozen printed books, it is usually still heavier than a single paperback volume today.

dimensions: The physical dimensions of an e-book are fixed, and cannot be changed individually for each title. Printed titles appear in a large variety of formats that often serve a certain purpose (e.g., pocketable paperback vs. large photography volume).

parallel use: To view several books next to each other requires several hardware devices. This is a fundamental problem which is only partially alleviated by windowing user interfaces. A solution would be multiple pages that can display digital ink, such as in Xerox' Gyricon or MIT's E-Ink project.

power consumption: Traditional paper books are practically always readable. To reduce the problem of having no access to material on an e-book because of a "dead battery", using solar power as additional energy source could be useful.

fragility: Similarly, e-books are still far more liable to damage when dropped, bent, or otherwise abused. Even in contact with water, paper books have a more graceful degradation than e-books.

flexibility: A paper newspaper is "soft" enough to be spread out on a crowded desk or otherwise uneven surface. E-book hardware, however, is rigid and nonflexible.

Haptic feedback

thickness: Visualizing the amount of pages that are behind or in front of the current page is relatively easy, but conveying the haptic feeling of this thickness is much more difficult to achieve. But this feeling is useful to remember where one stopped reading, or read an interesting quote last time. Also, book thickness is an important measure the user applies when trying to retrieve a title from a shelf, or select a suitable title among alternatives.

browsing: To quickly judge whether a book is suitable for buying, people tend to thumb through a book quickly. A lot of effort has gone into replicating this navigation metaphor on e-books. For example, Digital's Lectrice file format included a pre-rendered bitmap representation of each page to allow for quick page turns, a prerequisite for the thumbing navigation metaphor. But the input technique has to be adequate too, for example using pressure-sensitive sliding pads to mimic the browsing movement.

paper and print quality: Traditional books are only produced in an expensive form if the publisher expects them to be of a relatively high significance to the readership, resulting in a large sales volume. Consumers therefore use the production quality of a paper book as one parameter to also judge the quality of its contents, such as its lasting importance, or the level of editing and reviewing it has received. E-book titles cannot use this haptic quality indicator.

Issues such as copyright questions have been left out of this list because they are not directly related to the user experience, or HCI in general.

6 Interaction patterns for electronic book design

Whatever electronic books are going to look like, it is certain that they will have to put special emphasis on communicating information to the reader in an intuitive and effortless way. This puts special demands on the design of the human-computer interface of such devices. It is obvious that related traditional disciplines, such as typesetting, layout, and graphic design, but also more theoretical fields, such as cognitive psychology, have a lot of experience to contribute to the conception of this new medium. But it is difficult to carry this knowledge over to the new domain.

HCI deals with many aspects of communication. Not only is it trying to improve the communication between human and computer, but it also often stands at the borderline between various disciplines within a software project, includins software engineering, graphic design, and user representatives.

Still, one of the major limits that HCI has not managed to break yet lies in communication. It is difficult for user interface experts to communicate their experience and methods to other design team members. This leads to acceptance problems between these groups. They result from a lack of understanding of the paradigms, methods, and values of the other profession.

The major reason is that existing collections of expertise are mostly in the form of concrete rules for the traditional target medium. A more generative representation of this expertise is required.

Pattern languages in architecture

Pattern languages have proven to be an adequate medium to communicate design experience, even to people from different professions. They were originally introduced by architect Christopher Alexander in his books about urban architecture [2, 3]. To create architectures that better fit and adapt to the needs of their users, Alexander suggests creating a language of *patterns* that capture the essence of successful architectural solutions to recurring problems which are described as systems of competing forces: "As an element in the world, each pattern is a relationship between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain spatial configuration which allows these forces to resolve themselves. As an element of language, a pattern is an instruction, which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces, wherever the context makes it relevant." [2, p. 247].

A tpyical example of the more than 250 patterns Alexander describes in [3] is the *Street Cafe:* To solve the problem of creating an *Identifiable Neighborhood* with *Activity Nodes* and *Public Squares* (all these are names of other patterns defined in his text), the *Street Cafe* pattern can be applied: Several rooms of a cafe create a *Gradient of Privacy*, from a terrace with tables near the busy path, to more quiet rooms in the back. Newspapers in one corner make it an inviting place to pass some time, etc. The pattern solves a number of competing forces, e.g., of privacy and the desire to observe passers-by. It can be implemented using a number of smaller patterns, like *A Place to Wait, Sitting Wall,* and *Opening to the Street.*

Ultimately, a pattern language leads the designer from abstract, large-scale to concrete, small-scale design decisions through the links between its patterns.

Pattern languages in software engineering

The software engineering community was introduced to the Pattern concept in 1987, and it was in fact an HCI design case. Beck and Cunningham [7] built on Alexander's ideas to create a small pattern language capturing user interface design rules for Smalltalk applications, and presented their findings at an OOPSLA87 conference workshop. Using their language of just five patterns, several application domain experts were able to design their own user interfaces based on Smalltalk mechanisms without any prior knowledge of the language.

These essentially didactic possibilities may have been one of the major reasons why the software engineering community took up the idea of design patterns so eagerly, especially after Gamma, Helm, Johnson and Vlissides (frequently referred to as the Gang of Four) published their seminal book describing a collection of patterns for object-oriented software design [12]: It offered software engineering a new and practical way to communicate software design experience.

In general, a software design pattern is considered to be a proven solution of a recurring software engineering problem that balances the competing design constraints optimally for a certain type of situation.

In software design, as in architecture, patterns need to have certain qualities to distinguish them from other, simpler ways of describing problem solutions: They must

- carry a *name* that is as self-explanatory as possible,
- clearly define the *context* in which they can be applied,
- state the *forces* (parameters, or interests), that need to be balanced by the solution,
- describe a proven *solution* to a recurring problem,
- give *examples* where they have been applied successfully,
- be flexible and adaptive enough to *generate* solutions in varying contexts,
- and *reference* other, smaller-scale patterns to use synergistically, in order to create a solution as a whole of high quality.

That last item is especially important: Only when a comprehensive collection of patterns has been structured and interlinked in this way, it can become a pattern language.

Patterns in HCI

At first sight, it seems that the pattern idea has been picked up by the HCI community only recently, especially since the CHI'97 workshop [6]. Surprisingly, however, Alexanders ideas have been referenced by major HCI works much earlier. In 1988, Donald Norman, in his 'POET' book which is a standard text in many HCI courses [13], cited Alexander as a designer whose work had particularly influenced him. Another HCI 'classic', Apples Human Interface Guidelines [4], quotes Alexander's *A Pattern Language* as a seminal book about environmental design.

A more detailed adaption of Alexander's ideas was done by by the Utrecht School of Arts. In an overview of their interaction design curriculum, they outlined how they had adapted Alexanders approach to interaction design, "...using patterns to phrase guidelines in a consistent format that leaves room for subtleties." [5].

The CHI'97 workshop on Pattern Languages for Interaction Design [6] was an important event for the issue of pattern languages in HCI. It collected quite different opinions on how to carry the patterns idea over to HCI, for example from using patterns to describe observed user activities without judgment, to using them to capture HCI design practice. But in particular, the participants agreed that they "...felt we were at the very beginning of the enterprise of understanding the role and utility of pattern languages for interaction design." [6].

Within the last two years, the field has gained in momentum, and a number of proposals for pattern languages of HCI design issues have been published. A prominent example is J. Tidwell's pattern language for interaction design [17] that, in its present form, covers a substantial area of user interface design, but others have been presented, especially at the various PloP Conferences on Pattern Languages of Programming. A useful set of resources about interaction design patterns can be found at the Interaction Design Patterns Home Page, maintained by T. Erickson [10], who also presented his ideas of pattern languages as a *lingua franca* for interaction design [11].

Recently, the author participated in a ChiliPLoP'99 workshop about interaction patterns that arrived at the following definition:

> "An Interaction Pattern Language generates space/time interaction designs that create a system image close to the user's mental model of the task at hand, to make the human-computer interface as transparent as possible." [9]

The author organizes another workshop on this topic at INTERACT'99 in Edinburgh in September.

E-book interface design patterns

The author believes, however, that the concept of creating a pattern language can and should be applied not only to disciplines like architecture, software engineering, or general HCI. Pattern languages are a more universal model of structuring design knowledge. They can be used in any discipline that requires structured, creative work of some kind to be carried out. In particular, it should be possible to describe domains such as typography, typesetting, graphic design, cognitive psychology, and communication design as patterns of a language.

This observation leads us to a central idea:

> The format in which user interface design guidelines for electronic books are expressed should be that of a pattern language.

The advantages of such an approach are:

- A structured pattern language can lead the designer of an e-book from abstract user interface issues to concrete, low-level design guidelines. Abstract patterns address the design of the system as a whole, such as overall navigation metaphors, etc. Low-level patterns describe proven solutions of details in the user interface, for example, the form factor of single buttons. It can help to educate new designers in industry or academia.

- The patterns can represent our collective design experience and values about e-book interface design. The format has proven to be very readable (see Alexander's books as an example), even for non-professionals. This can make our design interests more understandable to members of a design team from other disciplines.

- The pattern collection gathered from experience in an e-book design project helps to create the design rationale.

- The pattern format can serve as a way to pass on this experience from project to project, similar to a "corporate memory" infrastructure in a company.

- The pattern language supplies an easy-to-remember vocabulary for communication about successful, reusable e-book interface design solutions across discipline boundaries in a flexible form.

This idea is best described by a simple example pattern, adopted from Jennifer Tidwell's collection [17]:

Name: *User's Annotations*

Examples: Handwritten comments on a page margin, or sticky notes in a book are examples of the principle in traditional, paper-based media.

Context: This pattern addresses the situation where the contents of an e-book are complex enough to require a simple summary in the user's words, or the user has an important related thought to an item in the text. Also, the user considers it likely that this part of the text will be consulted again in the future.

Problem: How can the user's hard-won understanding of the contents be preserved for later referral?

Forces: For knowledge that is needed only infrequently, it is easier to keep it "in the world" than "in the head" [13].

The users know better what is memorable to them than the e-book designer or title author (customization).

The e-book designer cannot predict in what context the contents will be of use by the variety of readers.

Users get a sense of ownership and control over the e-book by modifying and customizing it.

Spatial placement of own additions to a text make it easier to recognize, retrieve, and remember it later on.

Solution: *Provide a way for users to add their own comments and other annotations to the e-book.*

Let users place annotations physically close to where they are needed. Allow for simple sketches as well as text. Let them add private comments for their eyes only, and public comments for others to read, possibly with group distinctions. Save the annotations from session to session.

Positive and negative consequences: Annotations must be very easy to add, modify, and remove. Otherwise, they will not be used (bad examples abound in systems such as Windows Help, or Acrobat Exchange). If a clear distinction between e-book contents and annotations is not provided, users will fear to "destroy" the original version by annotations. It must be possible to revert to the original state of the e-book title at any time. It must also be clear what annotations are passed on if a book title is passed on to another reader.

References: Annotations can be saved as part of the e-book's *Remembered State* (this would be another pattern). More powerful annotation mechanisms could be supported by a *Revision History* mechanism, although ease-of-use must not be sacrificed.

7 Summary

The above example should have demonstrated that HCI design expertise for e-books can be expressed in a way easy to read, understand, and apply. It also shows that such patterns have already been expressed in collections as the one cited above, and can serve as a good starting point for an e-book design pattern language. Such a language could wrap our considerations for e-book user interfaces into a usable format.

The author believes that it will also be fruitful to come up with a definition of e-books, and a genre classification, to determine how the genre influences features and user interface issues for electronic books. This text has supplied some starting points for these tasks.

References

[1] Annette Adler, Anuj Gujar, Beverly L. Harrison, Kenton O'Hara, and Abigal Sellen. A diary study of work-related reading: Design implications for digital reading devices. In *Proc. CHI'98*, pages 241–248, 1998.

[2] Christopher Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.

[3] Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, 1977.

[4] Apple Computer. *Macintosh Human Interface Guidelines*. Addison-Wesley, 1992.

[5] Lon Barfield, Willie van Burgsteden, Ruud Lanfermeijer, Bert Mulder, Jurriënne Ossewold, Dick Rijken, and Philippe Wegner. Interaction design at the utrecht school of the arts. *SIGCHI Bulletin*, 26(3):49–79, 1994.

[6] Elisabeth Bayle, Rachel Bellamy, George Casaday, Thomas Erickson, Sally Fincher, Beki Grinter, Ben Gross, Diane Lehder, Hans Marmolin, Brian Moore, Colin Potts, Grant Skousen, and John Thomas. Putting it all together: Towards a pattern language for interaction design. *SIGCHI Bulletin*, 30(1):17–23, 1998.

[7] Kent Beck and Ward Cunningham. Using pattern languages for object-oriented programs. Technical Report CR-87-43, Tektronix, Inc., September 17, 1987. Presented at the OOPSLA'87 workshop on Specification and Design for Object-Oriented Programming.

[8] Jan Borchers, Oliver Deussen, and Clemens Knörzer. Getting it across: Layout issues for kiosk systems. *SIGCHI Bulletin*, 27(4):68–74, 1995.

[9] Jan Borchers, Bill Brook (organizer), Andrew Carlson, Jens Coldewey, Todd Coram, and Anthony Flaks. Interaction patterns workshop. Presentation at ChiliPLoP'99 Conference on Pattern Languages of Programming. http://www.interaction-patterns.org, March 16–19 1999.

[10] Thomas Erickson. Interaction patterns home page. Established February 1998. http://www.pliant.org/personal/Tom_Erickson/InteractionPatterns.html.

[11] Thomas Erickson. Interaction pattern languages: A *Lingua Franca* for interaction design? In *Usability Professionals' Association Conference '98 (invited talk)*, Washington, D.C., June 24 1998. http://www.upassoc.org/html/download/patterns.ppt.

[12] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA, 1995.

[13] Donald A. Norman. *The Psychology of Everyday Things*. Basic Books, New York, 1988.

[14] David J. Oborne and Doreen Holton. Reading from screen versus paper: There is no difference. *International Journal of Man-Machine Studies*, 28:1–9, 1988.

[15] Jürgen Rink. Die Geister, die ich rief: Chancen und Risiken der elektronischen Bücher. *c't*, 6:192–202, 1999. (In German).

[16] Ben Shneiderman. *Designing the User Interface (Third Edition)*. Addison-Wesley, 1998.

[17] Jennifer Tidwell. Interaction design patterns. PLoP'98 Conference on Pattern Languages of Programming, Illinois, extended version at http://www.mit.edu/~jtidwell/interaction_patterns.html, 1998.

DESIGN PATTERNS The book is then dedicated to document in extensive detail using visual examples and pointing out differences across platforms and/or interaction constraints. Each pattern consists of the following sections: 1) Problem - the situation being addressed through design (i.e. you want to display a list of data to the user) 2) Solution - the definition of the specific pattern (i.e. Vertical List, Scrolling, etc.) 3) Variations - a list of similar patterns 4) Interaction Details - a description of the actual interaction 5) Presentation Details - a visual representation of the pattern By capturing UI best practices and reusable ideas as design patterns, Designing Interfaces provides solutions to common design problems that you can tailor to the situation at hand. Designing Interfaces: Patterns for Effective Interaction Design is an intermediate-level book about interface and interaction design, structured as a pattern language. It features real-live examples from desktop applications, web sites, web applications, mobile devices, and everything in between. This site contains excerpts from some of the book's patterns. The book has more, of course -- more introductory mat Interaction design pattern. From Wikipedia, the free encyclopedia. A design pattern is a formal way of documenting a solution to a common design problem.Â Design patterns gained popularity in computer science after the book Design Patterns: Elements of Reusable Object-Oriented Software was published. Since then a pattern community has emerged that specifies patterns for problem domains including architectural styles and object-oriented frameworks.Â Applying a pattern language approach to interaction design was first suggested in Norman and Draper's book User Centered System Design (1986). The Apple Computer's Macintosh Human Interface Guidelines also quotes Christopher Alexander's works in its recommended reading. Libraries.